Visão Computacional e Processamento de Imagens

Prof. André Gustavo Hochuli

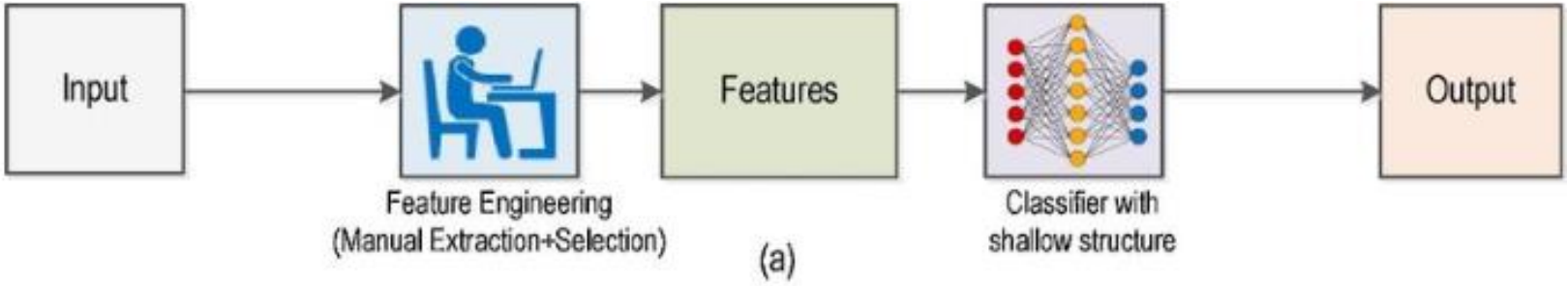
gustavo.hochuli@pucpr.br
aghochuli@ppgia.pucpr.br

Tópicos

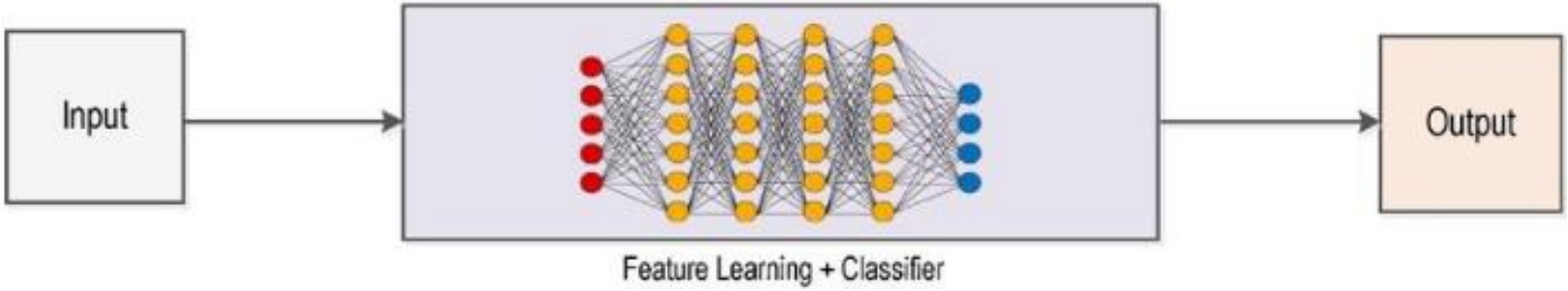
- Revisão Aula 01
  - Visão Computacional
  - Definições de Imagem
  - Sistemas de Cor
- Convolução e Filtros
- Extração de Componentes
- Descritores de Características (Introdução)

Visão Computacional Workflow

- **Abordagem Tradicional (~2010)**

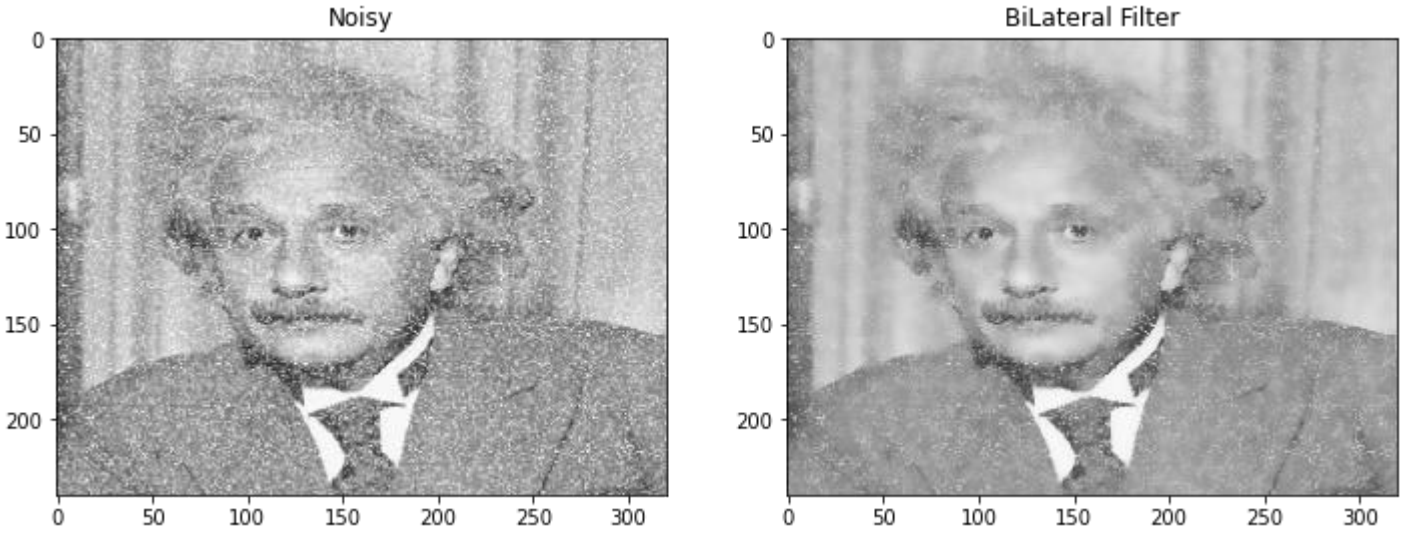


- **Deep Learning (~2010->Hoje)**



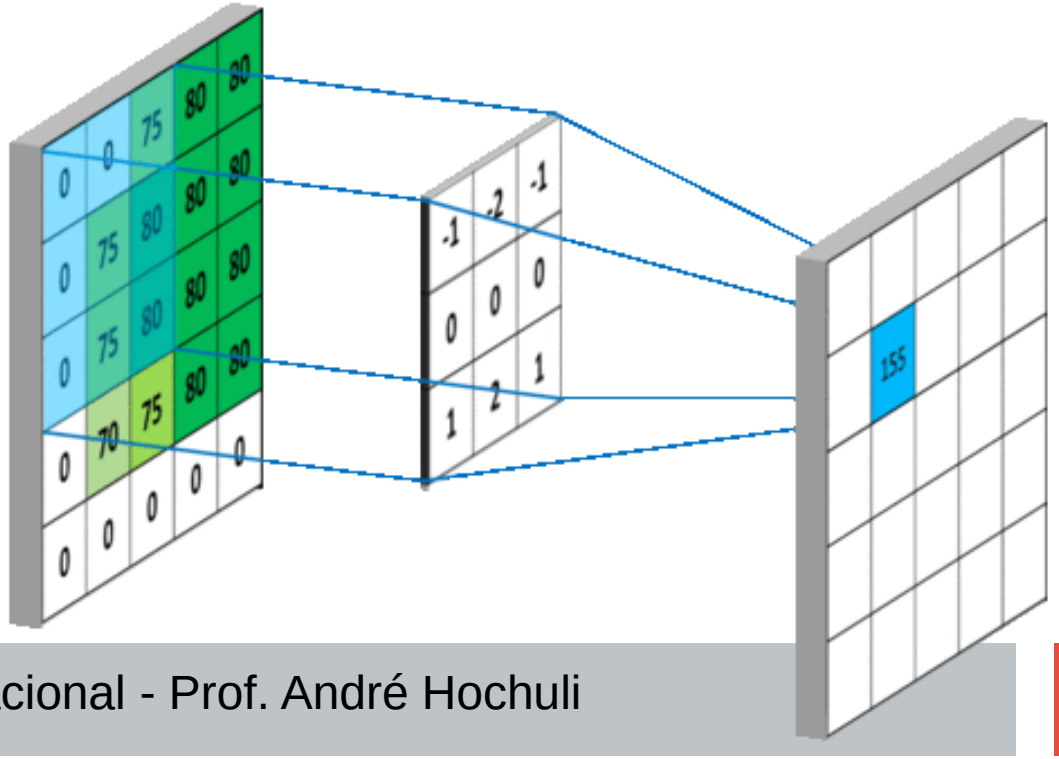
Filtros de Imagem

- **Aprimoramento de Imagens**
- **Redução de Ruídos**
- **Operações Matemáticas**



Convolução

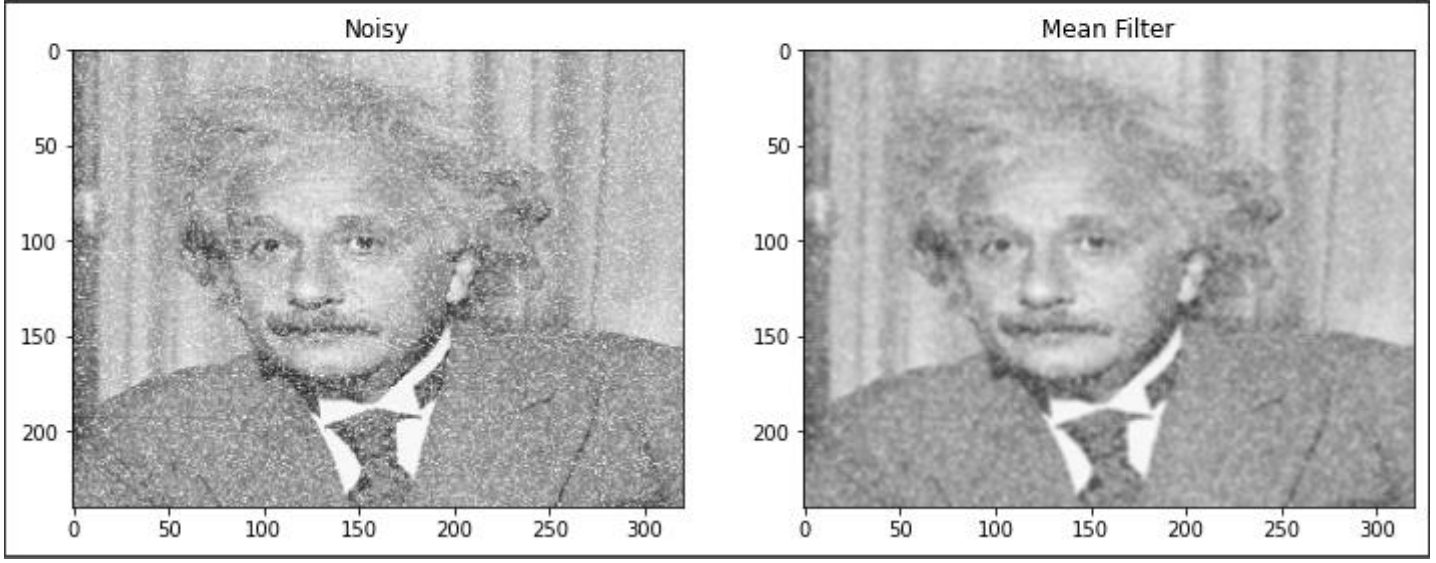
- Deslizar um Kernel (Filtro Convolucional) sobre a imagem
  - Stride, Kernel Size, Weigths
- Transforma o pixel central a partir dos pesos do kernel sobre a vizinhança do pixel



Filtro da Média

- **Substitui o pixel central pela sua média**
- **Espalha o ruído (outlier) pela vizinhança**
- **Detalhes são suavizados**

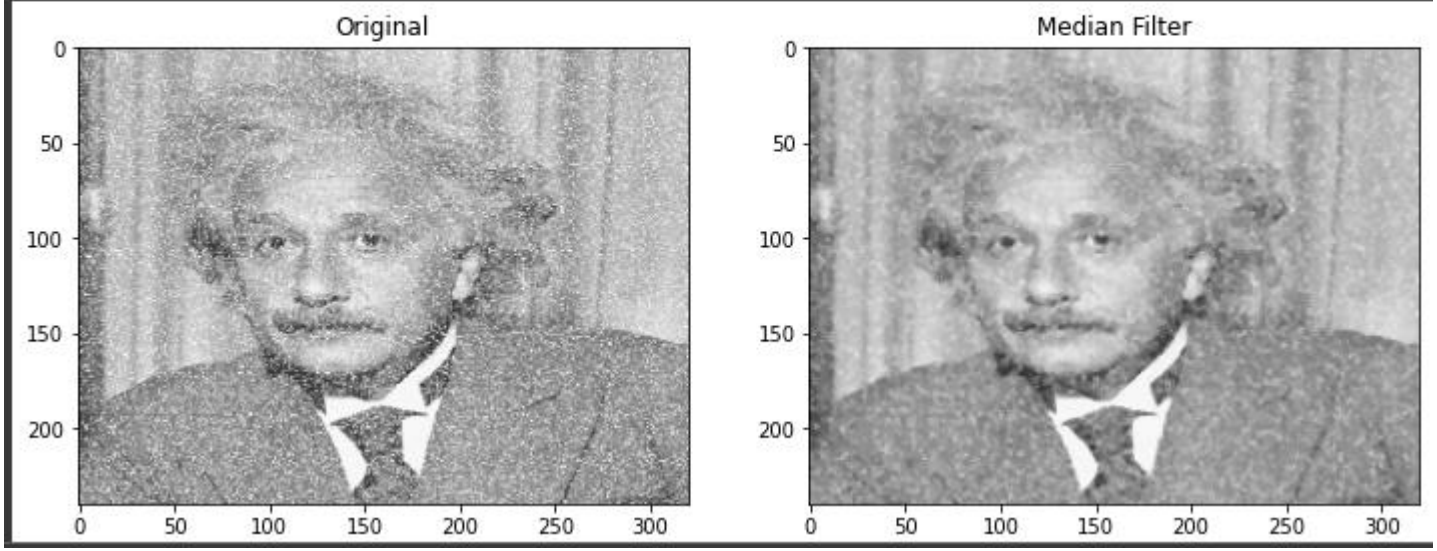
```
kernel_size = (3,3)
im_blur = cv2.blur(im_einstein,kernel_size)
plot_sidebyside([im_einstein,im_blur],['Noisy','Mean Filter'],colormap='gray')
```



Filtro da Mediana

- **Substitui o pixel central pela mediana vizinhança**
- **Preserva melhor os detalhes do que comparado a média**

```
im_blur = cv2.medianBlur(im_einstein,3)
plot_sidebyside([im_einstein,im_blur],['Noisy','Median Filter'],colormap='gray')
```

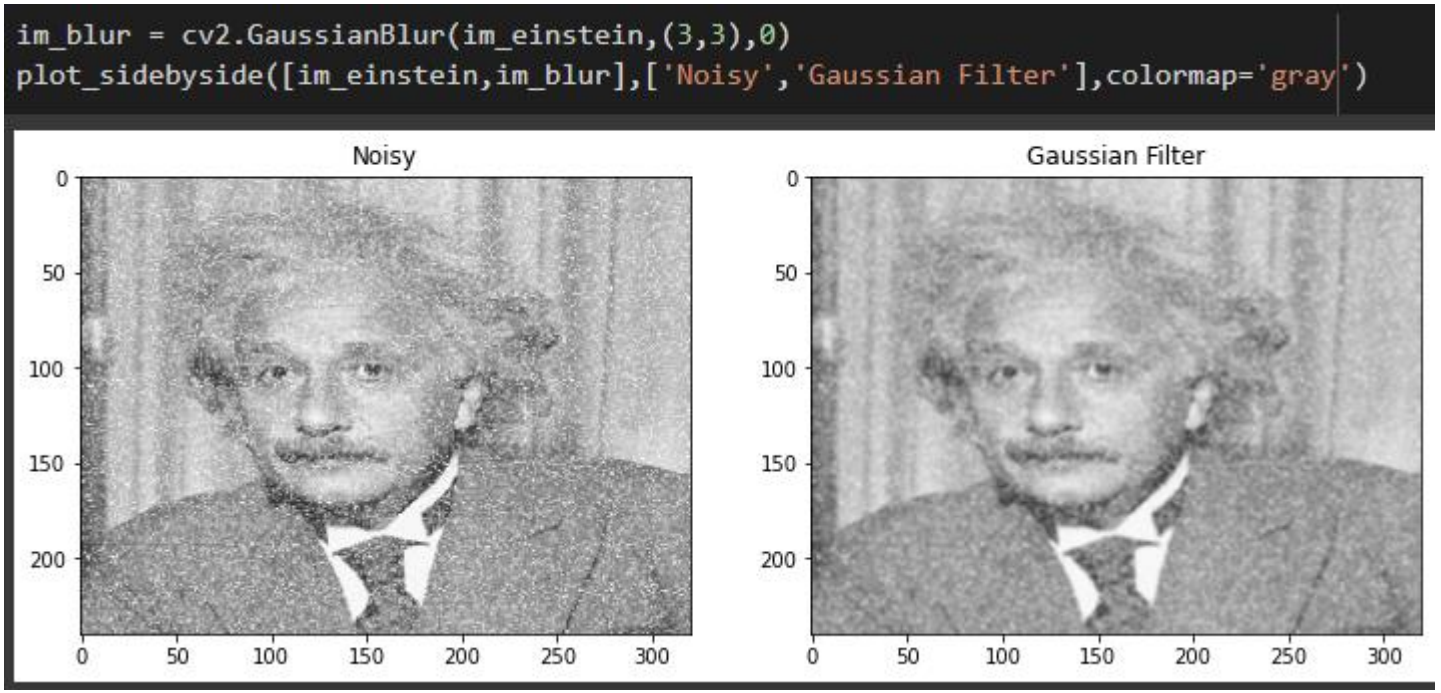


Filtro Gaussiano

- **Distribuição Gaussiana dos Pixels**
- **Os pesos são compostos por distribuições**
- **Media Ponderada**
- **Desvio padrão determina o grau do filtro.**

1/16	1/8	1/16
1/8	1/4	1/8
1/16	1/8	1/16

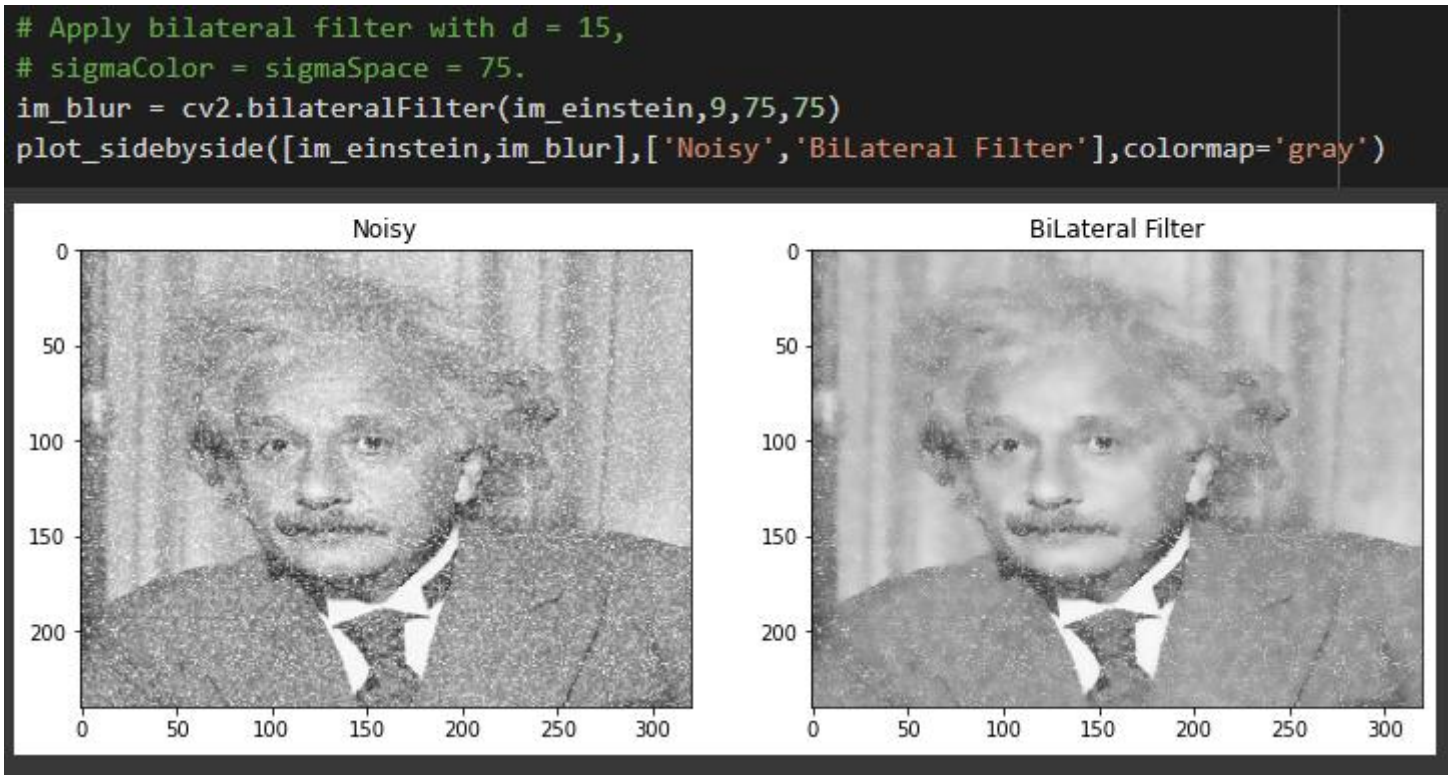
```
im_blur = cv2.GaussianBlur(im_einstein,(3,3),0)
plot_sidebyside([im_einstein,im_blur],['Noisy','Gaussian Filter'],colormap='gray')
```



Filtro Bilateral

- **Baseado em distribuição gaussiana**
- **Normalização dos fatores**
- **Preserva detalhes**

```
# Apply bilateral filter with d = 15,
# sigmaColor = sigmaSpace = 75.
im_blur = cv2.bilateralFilter(im_einstein,9,75,75)
plot_sidebyside([im_einstein,im_blur],['Noisy','BiLateral Filter'],colormap='gray')
```



Let’s Code!

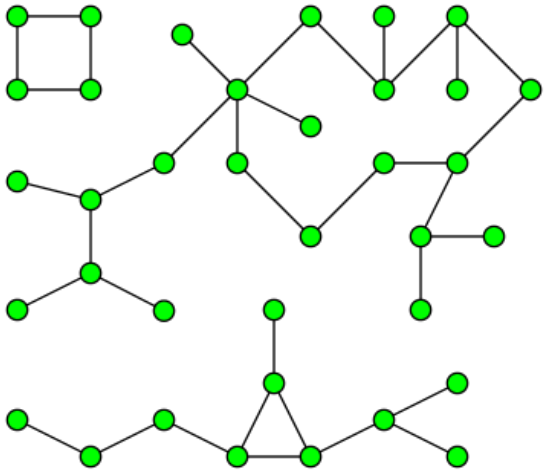
- **Siga o [LINK]**

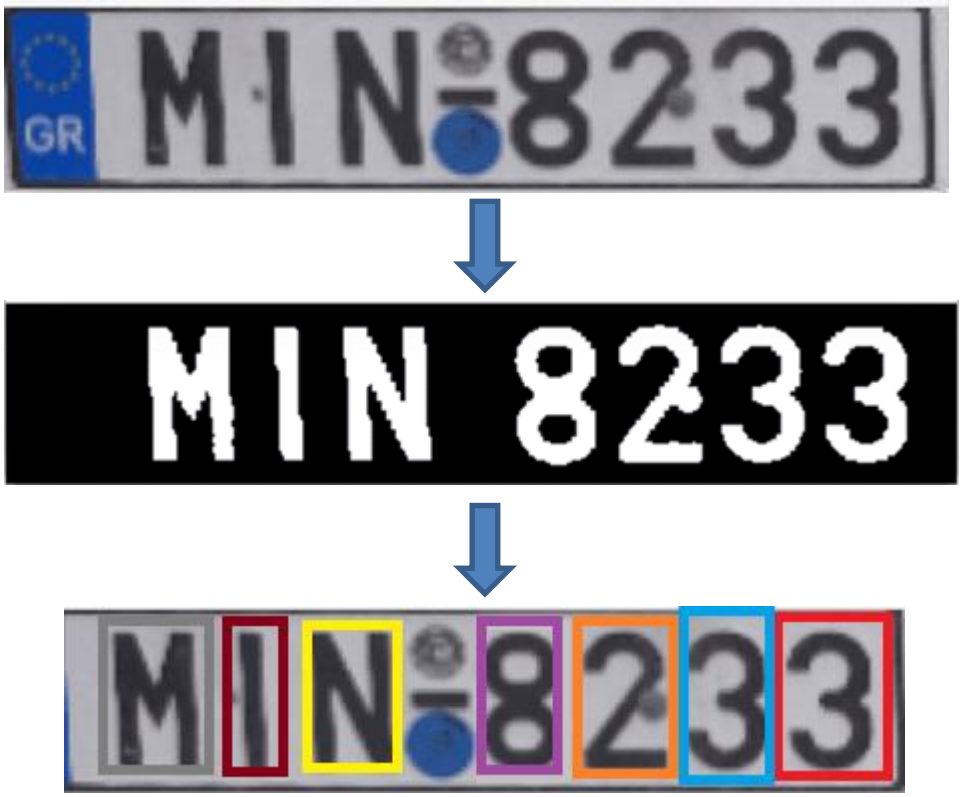
Detecção de Componentes Conexos

Segmentação de Componentes

- Componentes Conexos

- Aplicação oriunda em teoria de grafos
  - Redes Sociais
  - Biologia
  - Reconhecimento de Padrões

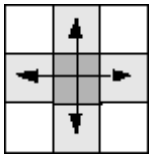




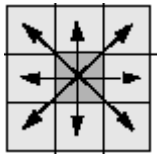
Algoritmo de Detecção Clássico

- Analisa a vizinhaça (foreground)
- Rotula cada componente candidato não-conectado (1,2,3,4….)

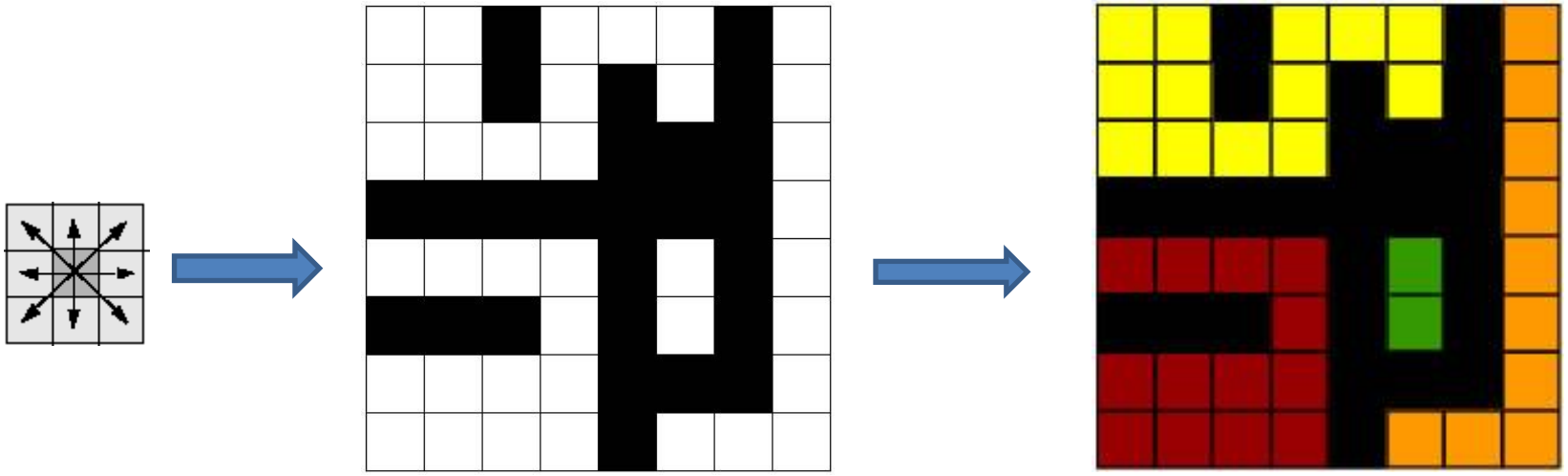
- Kernels:



4-Neighboors



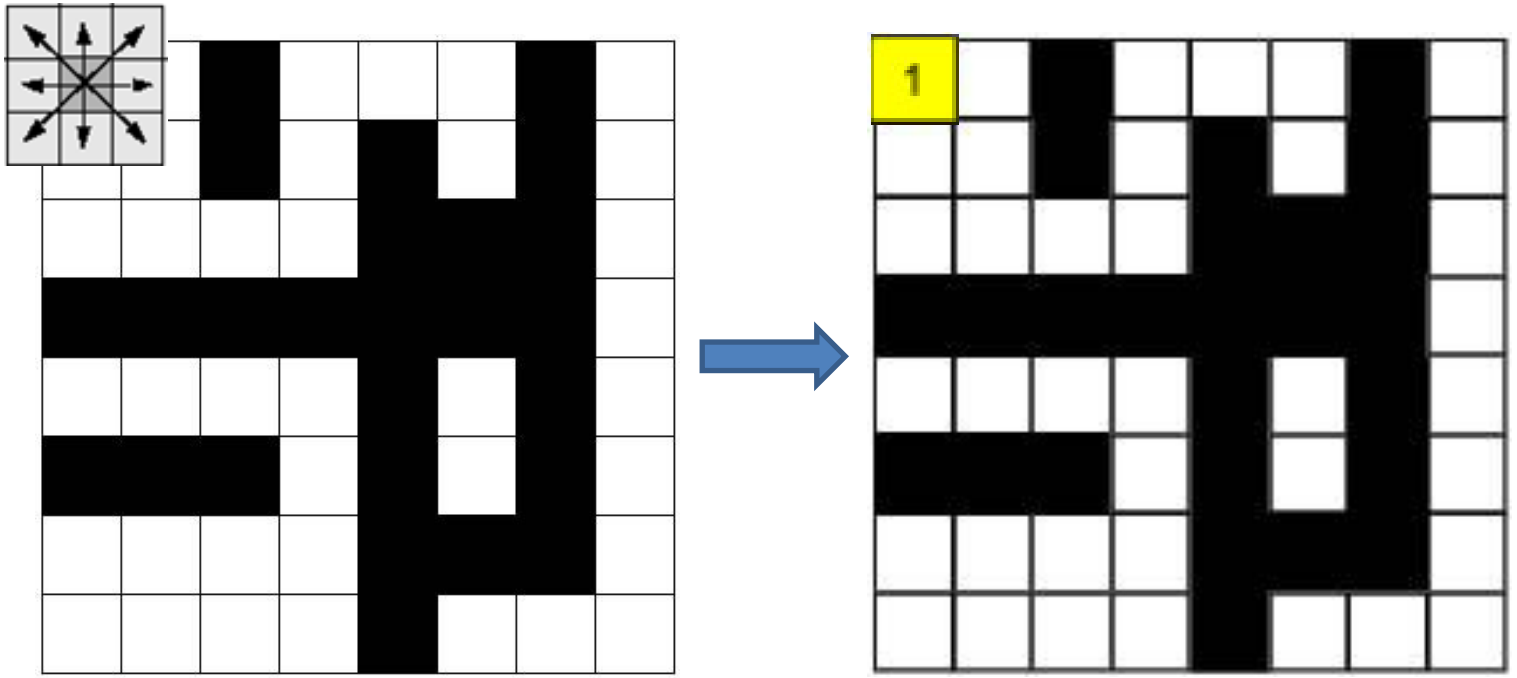
8-Neighboors



Linha a Linha

- Deslizar o kernel , linha a linha (2 passes)
  - Se o centro for *Foreground* rotula!
  - Rotulação:
    - Se não tem pixels conexos, atribui um novo rotulo
    - Senão, replica-se o rotulo do vizinho
    - Um estrutura (Union-Find) controla as adjacencias

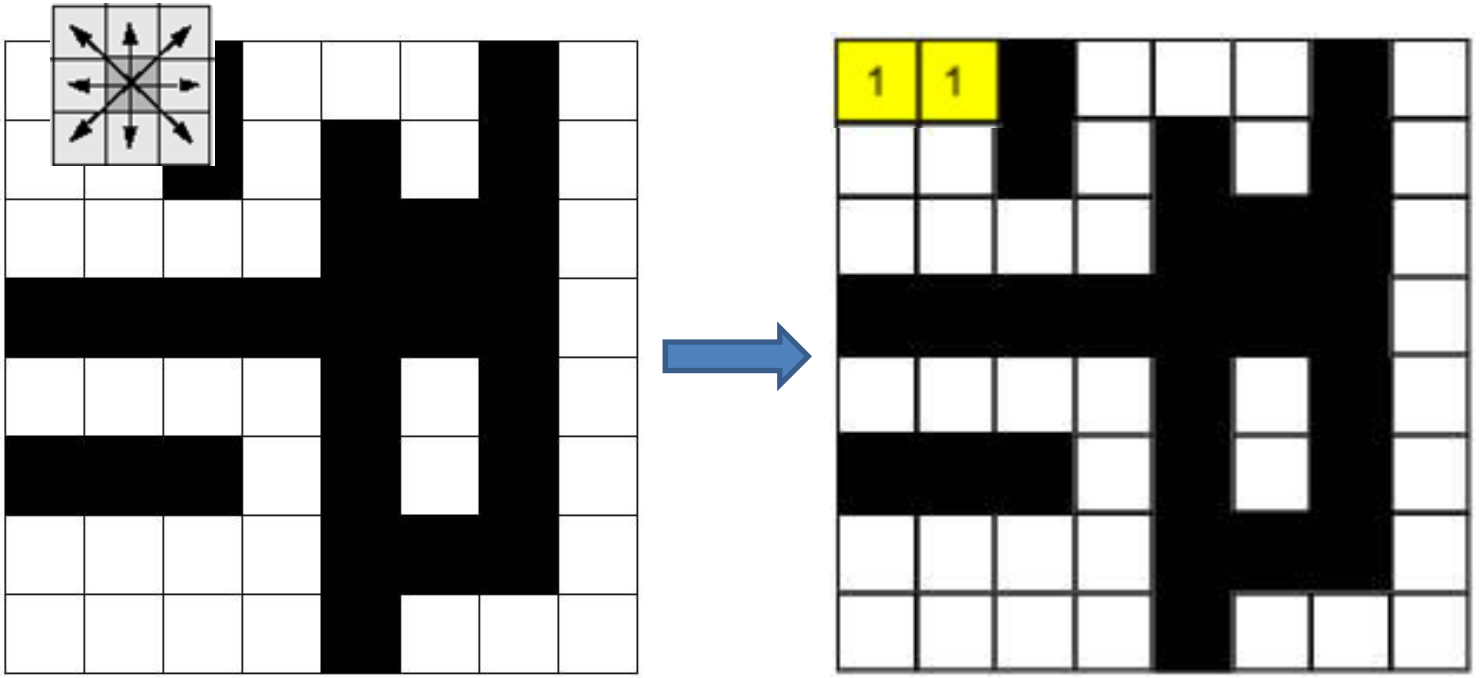
- **Pass #1:**
  - **Row #1**



Linha a Linha

- Deslizar o kernel , linha a linha (2 passes)
  - Se o centro for *Foreground* rotula!
  - Rotulação:
    - Se não tem pixels conexos, atribui um novo rotulo
    - Senão, replica-se o rotulo do vizinho
    - Um estrutura (Union-Find) controla as adjacencias

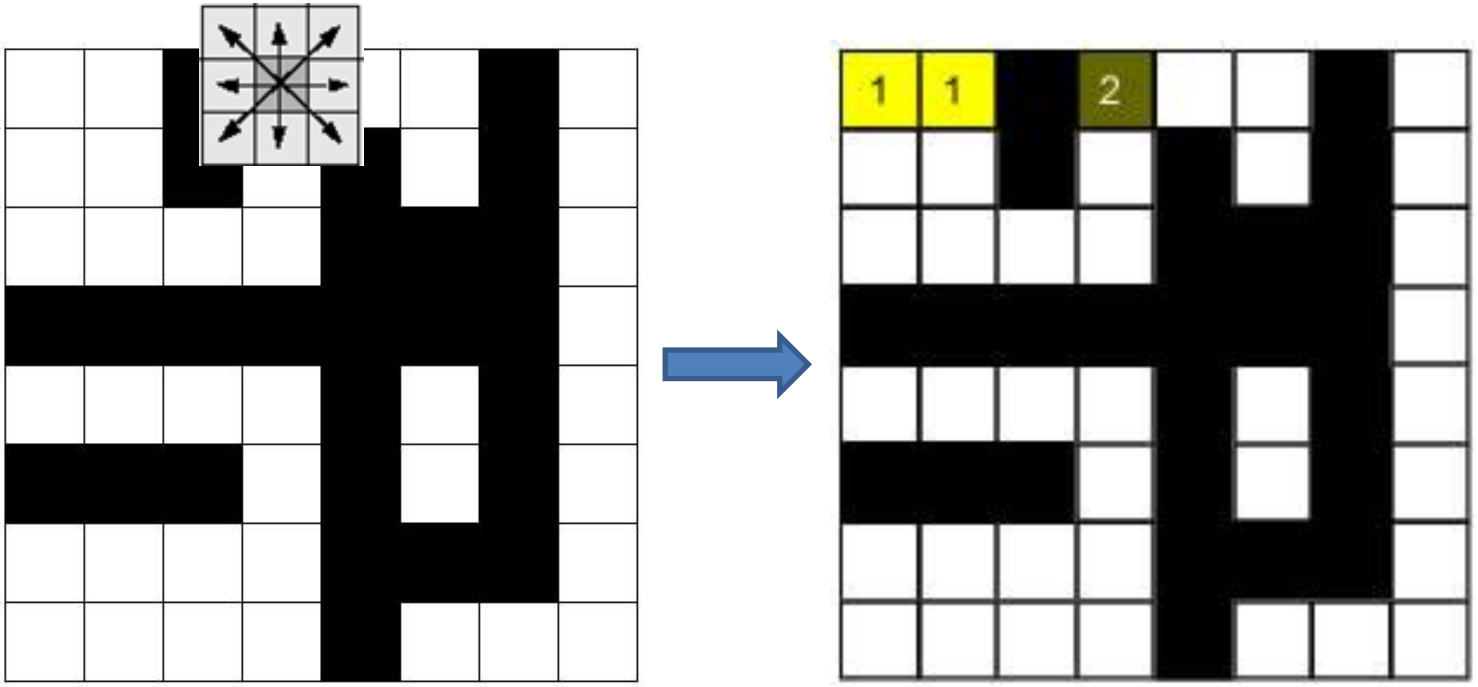
- **Pass #1:**
  - **Row #1**



Linha a Linha

- Deslizar o kernel , linha a linha (2 passes)
  - Se o centro for *Foreground* rotula!
  - Rotulação:
    - Se não tem pixels conexos, atribui um novo rotulo
    - Senão, replica-se o rotulo do vizinho
    - Um estrutura (Union-Find) controla as adjacencias

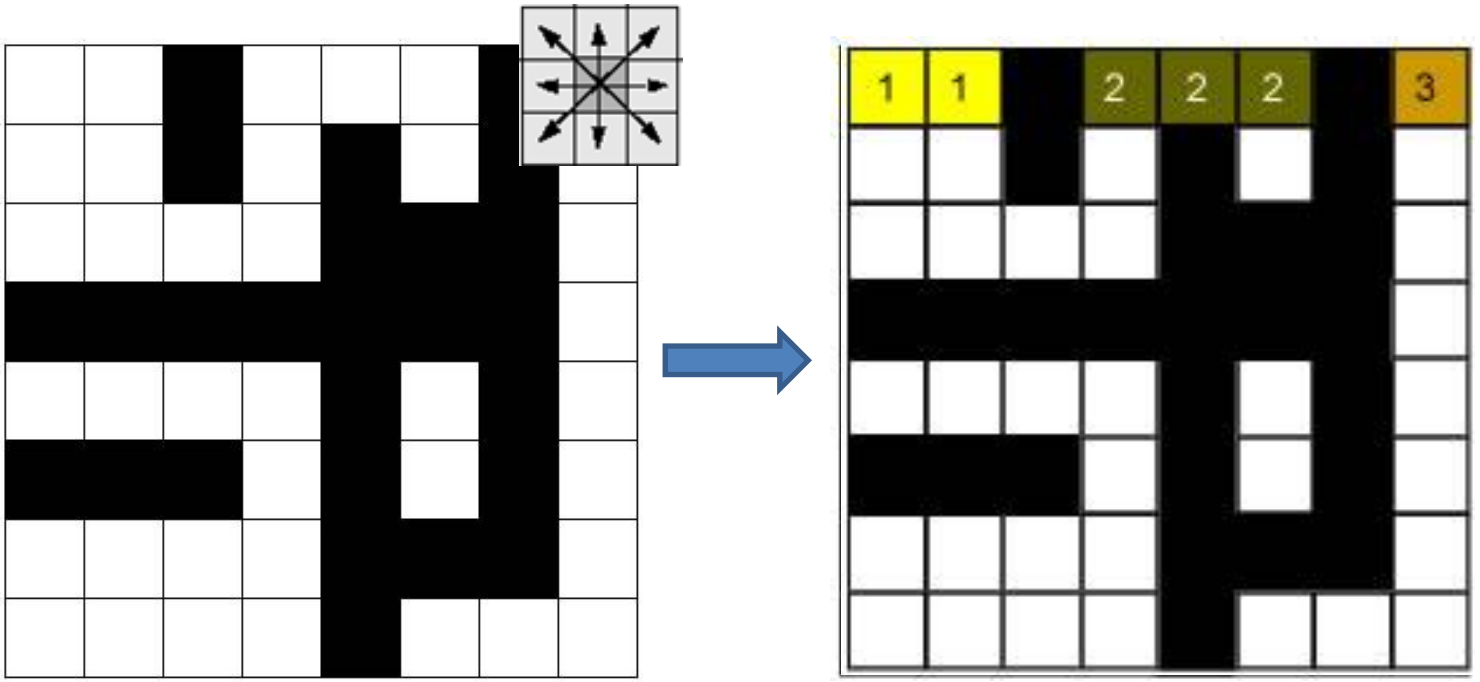
- **Pass #1:**
  - **Row #1**



Linha a Linha

- Deslizar o kernel , linha a linha (2 passes)
  - Se o centro for *Foreground* rotula!
  - Rotulação:
    - Se não tem pixels conexos, atribui um novo rotulo
    - Senão, replica-se o rotulo do vizinho
    - Um estrutura (Union-Find) controla as adjacencias

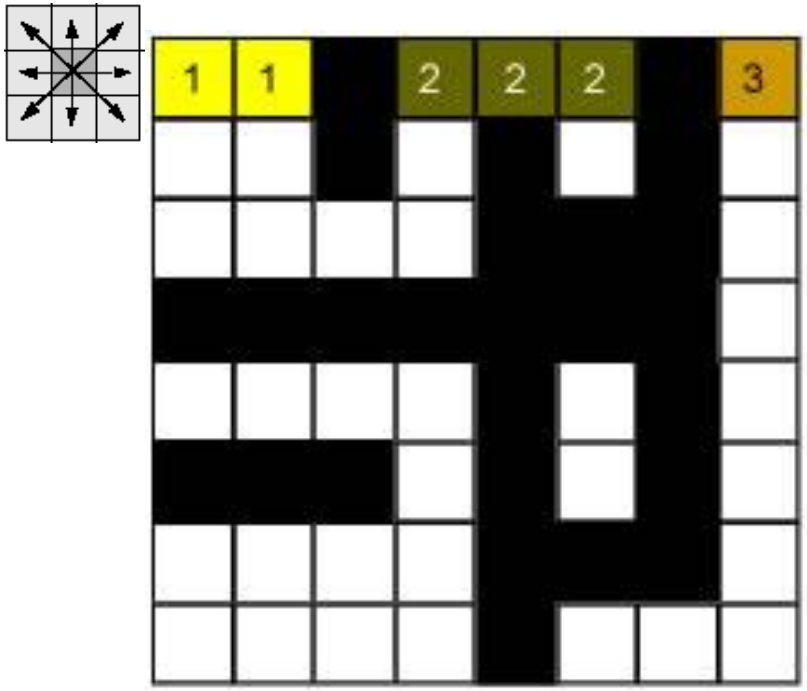
- **Pass #1:**
  - **Row #1**



Linha a Linha

- Deslizar o kernel , linha a linha (2 passes)
  - Se o centro for *Foreground* rotula!
  - Rotulação:
    - Se não tem pixels conexos, atribui um novo rotulo
    - Senão, replica-se o rotulo do vizinho
    - Um estrutura (Union-Find) controla as adjacencias

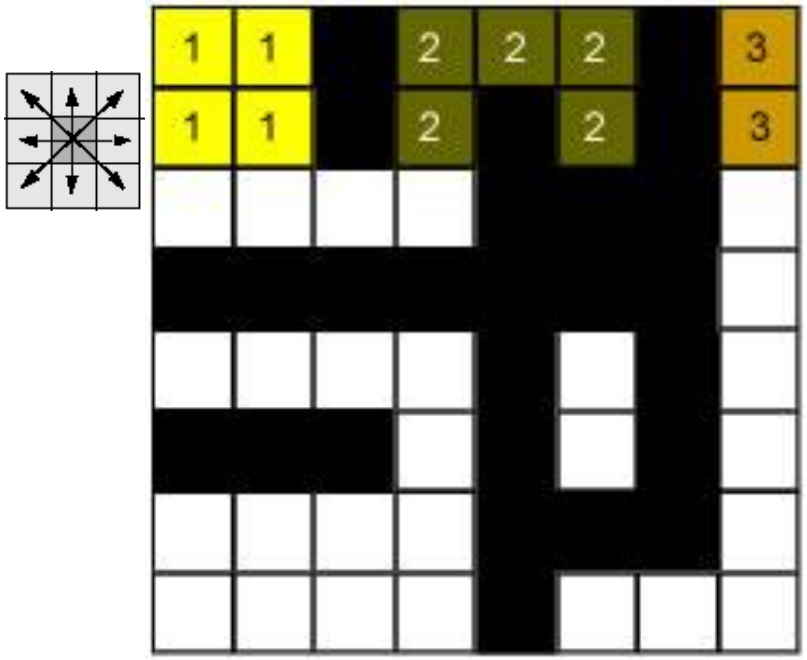
- **Pass #1:**
  - **Row #1**



Linha a Linha

- Deslizar o kernel , linha a linha (2 passes)
  - Se o centro for *Foreground* rotula!
  - Rotulação:
    - Se não tem pixels conexos, atribui um novo rotulo
    - Senão, replica-se o rotulo do vizinho
    - Um estrutura (Union-Find) controla as adjacencias

- **Pass #1:**
  - **Row #1**



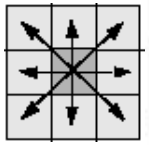
Linha a Linha

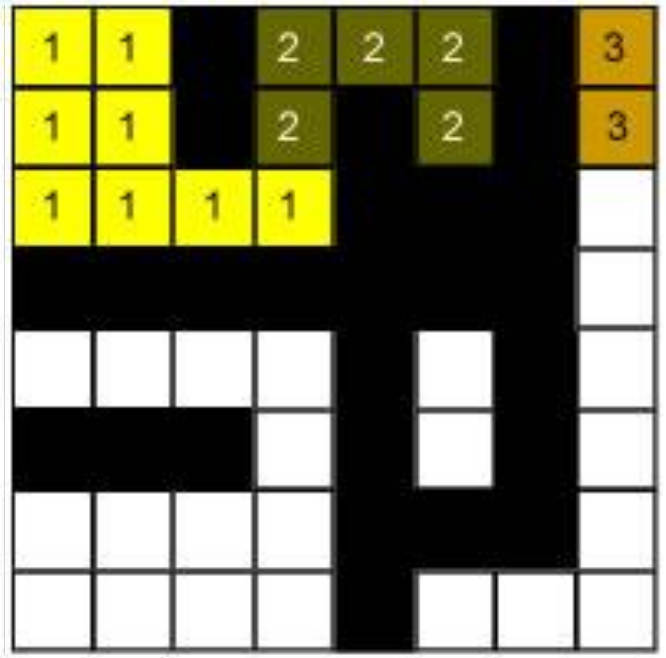
- Sliding a connectivity kernel , row by row (2 passes)
  - If the center falls in a non-zero pixel, label it!
  - Labeling:
    - If there are no labeled pixels connected, attribute a new label
    - Otherwise, attribute to it the neighbor´s label.
    - A Union-Find structure control adjacent labels (Union-Find)

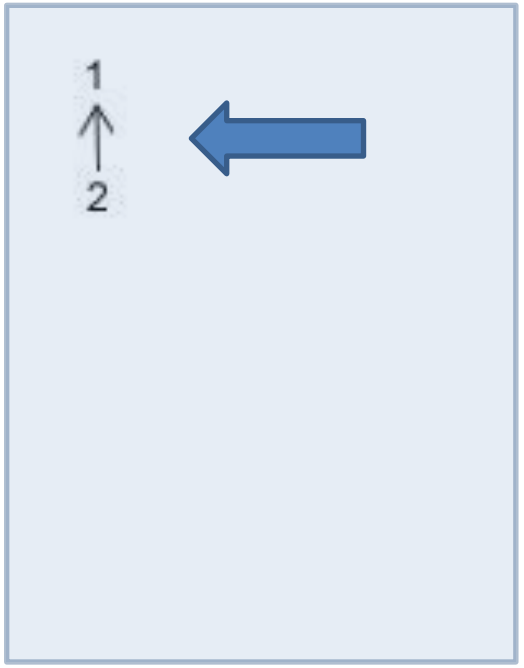
Adjacent labels

Union-Find

- **Pass #1:**
  - **Row #3**



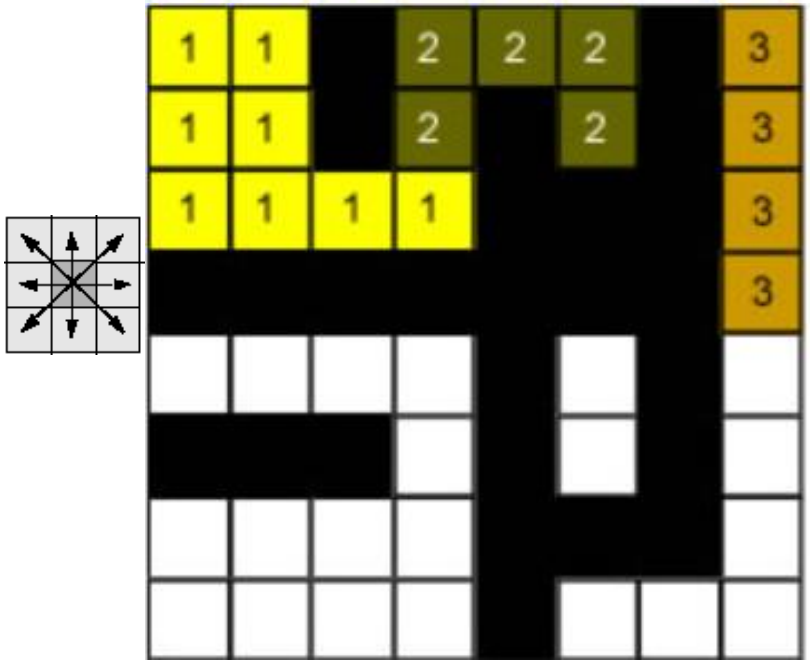




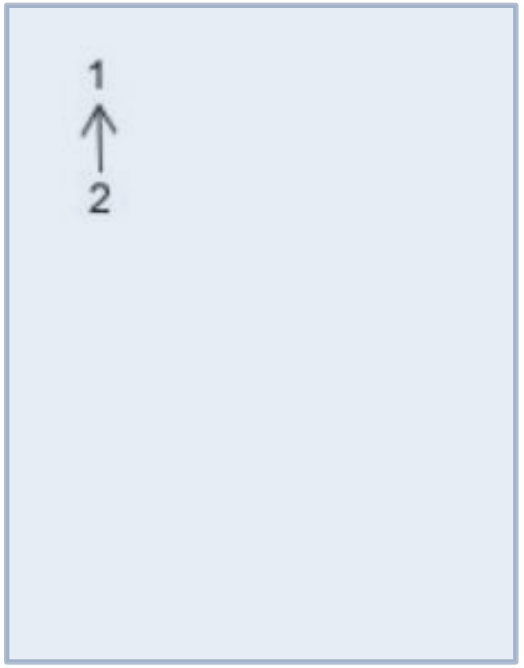
Linha a Linha

- Deslizar o kernel , linha a linha (2 passes)
  - Se o centro for *Foreground* rotula!
  - Rotulação:
    - Se não tem pixels conexos, atribui um novo rotulo
    - Senão, replica-se o rotulo do vizinho
    - Um estrutura (Union-Find) controla as adjacencias

- **Pass #1:**
  - **Row #1**



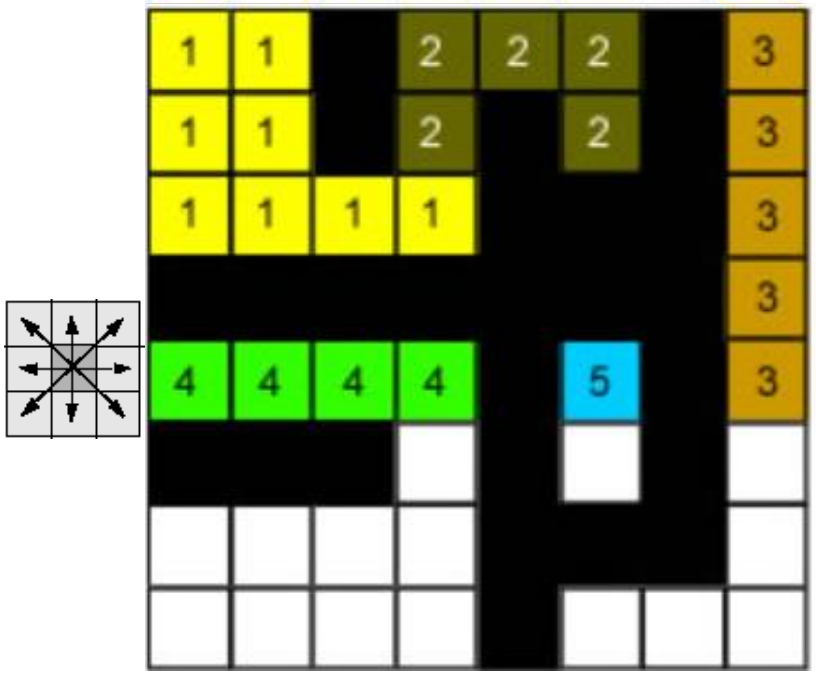
Union-Find



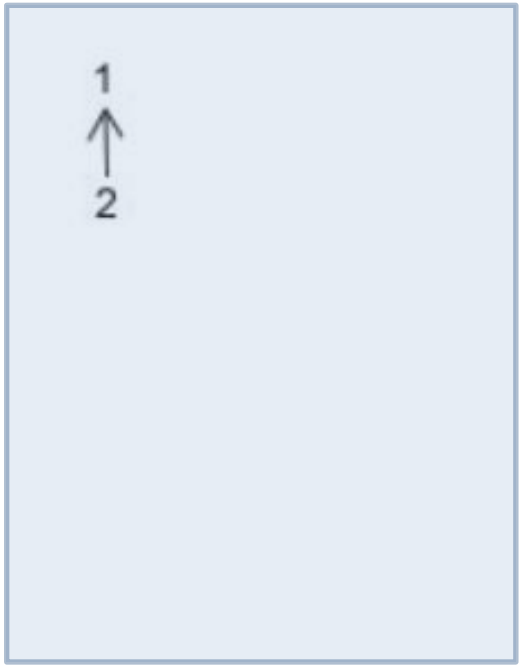
Linha a Linha

- Deslizar o kernel , linha a linha (2 passes)
  - Se o centro for *Foreground* rotula!
  - Rotulação:
    - Se não tem pixels conexos, atribui um novo rotulo
    - Senão, replica-se o rotulo do vizinho
    - Um estrutura (Union-Find) controla as adjacencias

- **Pass #1:**
  - **Row #1**



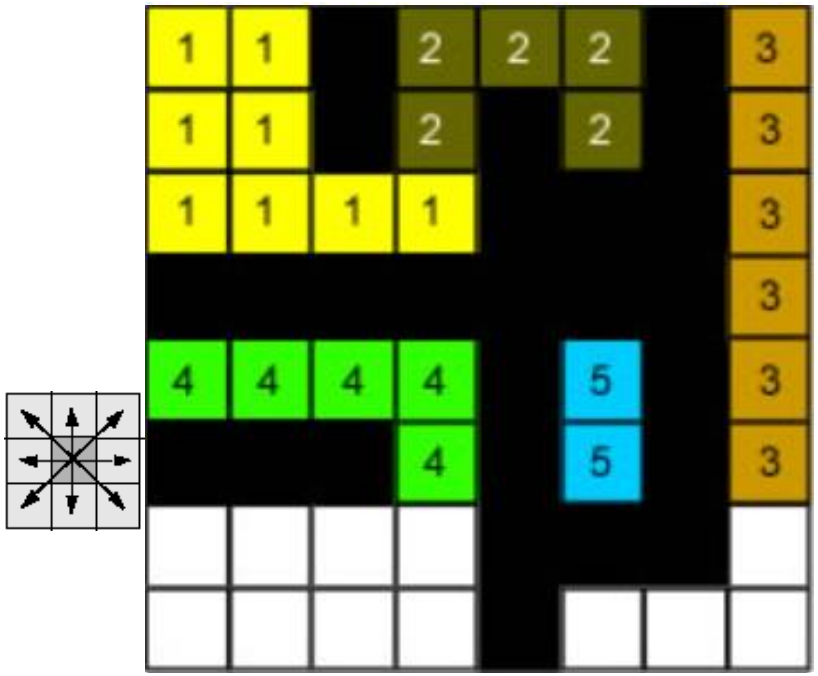
Union-Find



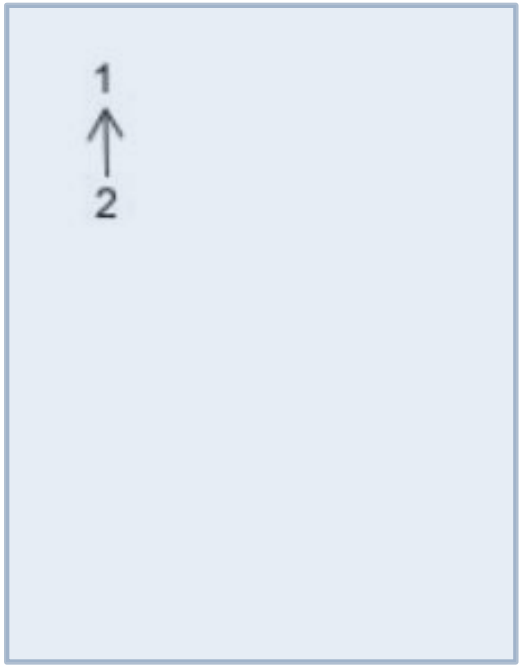
Linha a Linha

- Deslizar o kernel , linha a linha (2 passes)
  - Se o centro for *Foreground* rotula!
  - Rotulação:
    - Se não tem pixels conexos, atribui um novo rotulo
    - Senão, replica-se o rotulo do vizinho
    - Um estrutura (Union-Find) controla as adjacencias

- **Pass #1:**
  - **Row #1**



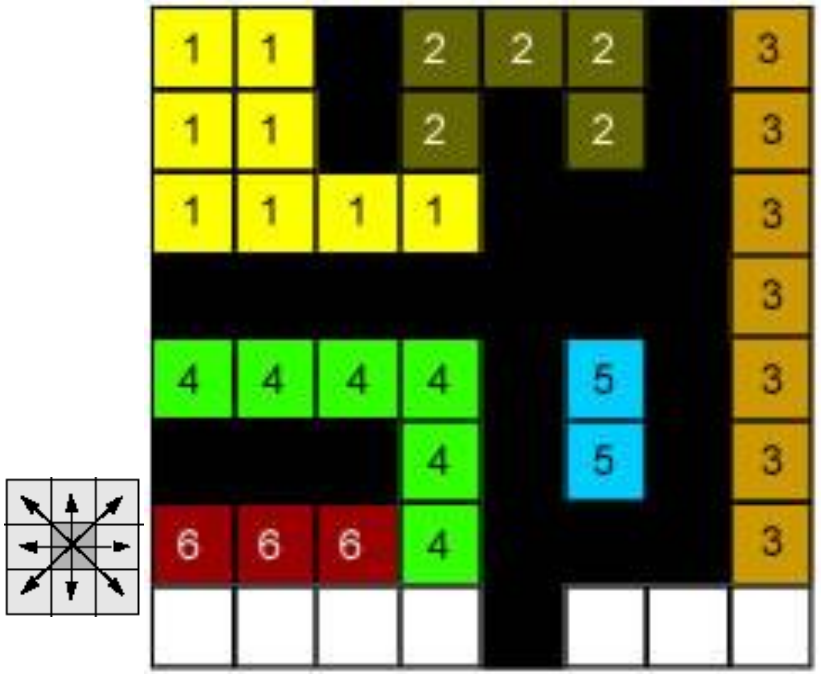
Union-Find



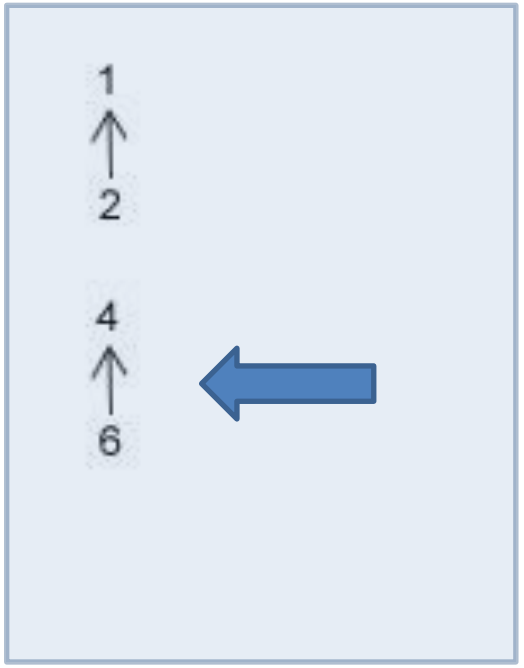
Linha a Linha

- Deslizar o kernel , linha a linha (2 passes)
  - Se o centro for *Foreground* rotula!
  - Rotulação:
    - Se não tem pixels conexos, atribui um novo rotulo
    - Senão, replica-se o rotulo do vizinho
    - Um estrutura (Union-Find) controla as adjacencias

- **Pass #1:**
  - **Row #1**



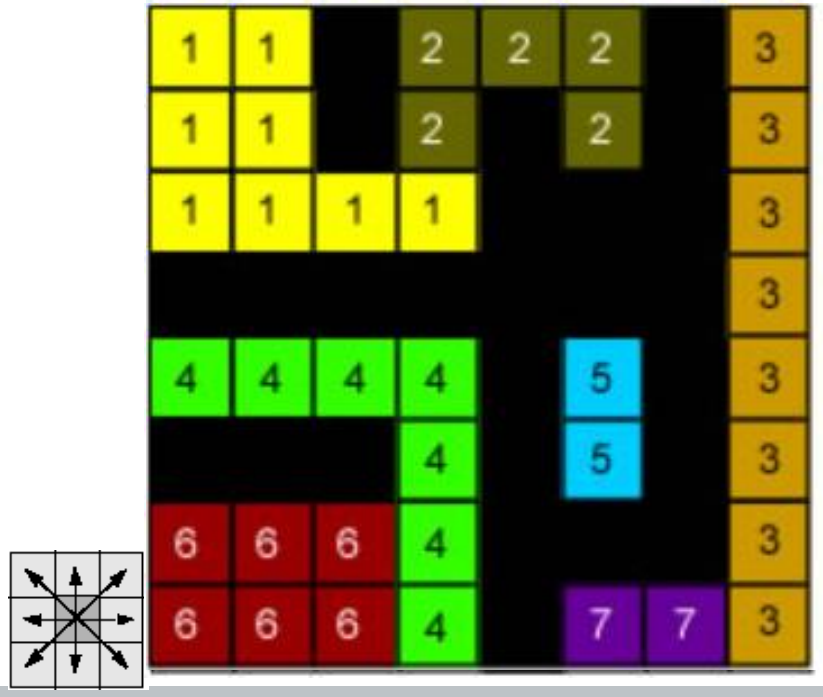
Union-Find



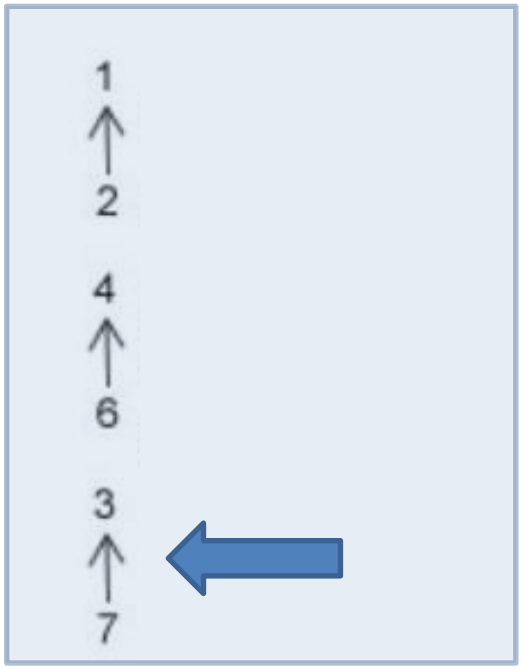
Linha a Linha

- Deslizar o kernel , linha a linha (2 passes)
  - Se o centro for *Foreground* rotula!
  - Rotulação:
    - Se não tem pixels conexos, atribui um novo rotulo
    - Senão, replica-se o rotulo do vizinho
    - Um estrutura (Union-Find) controla as adjacencias

- **Pass #1:**
  - **Row #1**



Union-Find

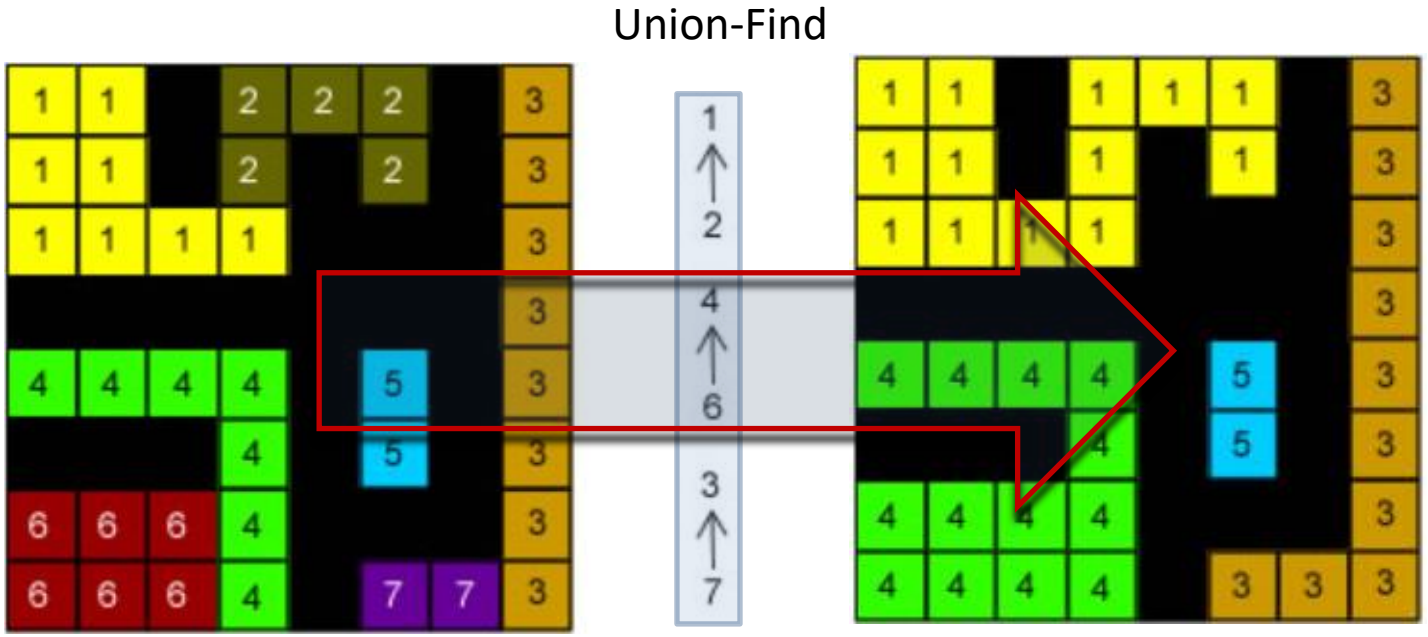


Linha a Linha

- Deslizar o kernel , linha a linha (2 passes)
  - Se o centro for *Foreground* rotula!
  - Rotulação:
    - Se não tem pixels conexos, atribui um novo rotulo
    - Senão, replica-se o rotulo do vizinho
    - Um estrutura (Union-Find) controla as adjacencias

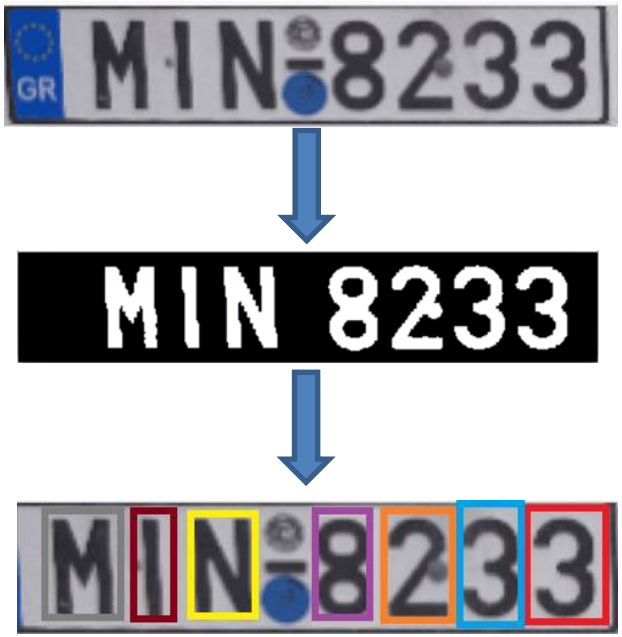
- **Pass #2:**

- **Resolve Union-Find**



Let's Code!

- Vamos implementar um segmentar caracteres de placas automotivas

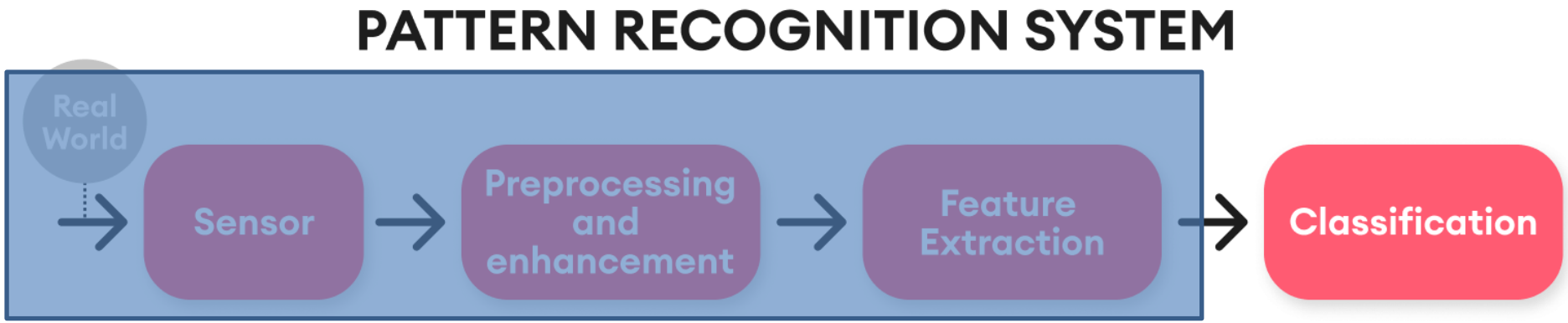


- Também vamos introduzir a função cv2.connectedComponent() que implemente o algoritmo de componente conexo

- Siga o [LINK]

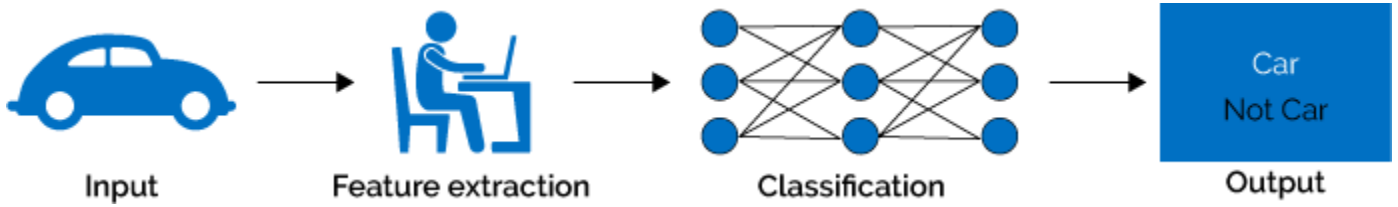
Extração de Características

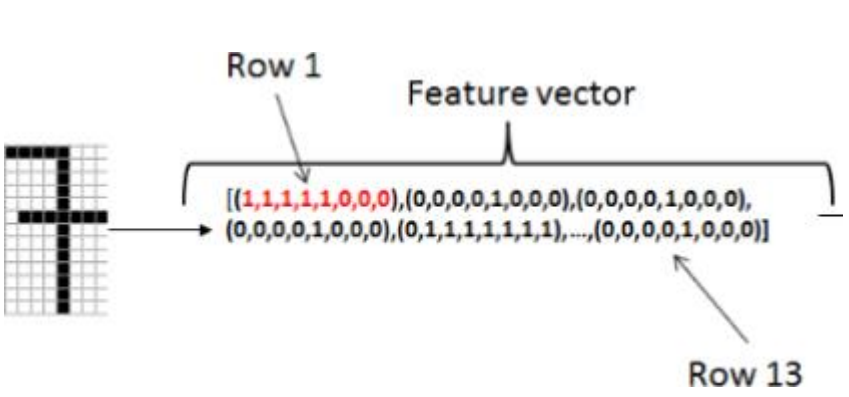
Visão Computacional Workflow

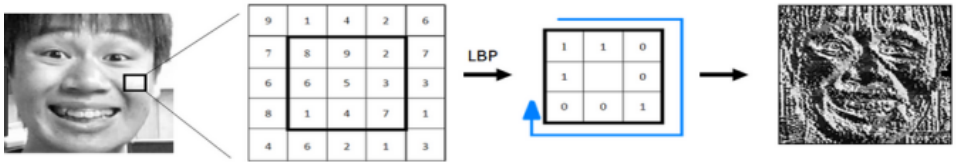


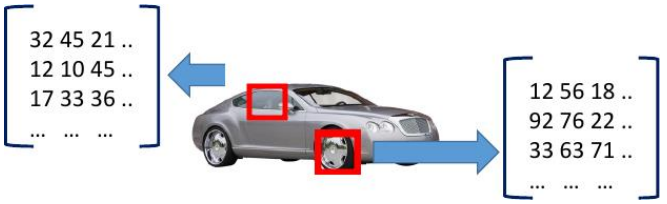
Extração de Características

- Descritor de característica converte um dado de alta dimensão em um espaço de característica
- Um vetor de característica representa o dado
- Então, um modelo computacional aprende a representação



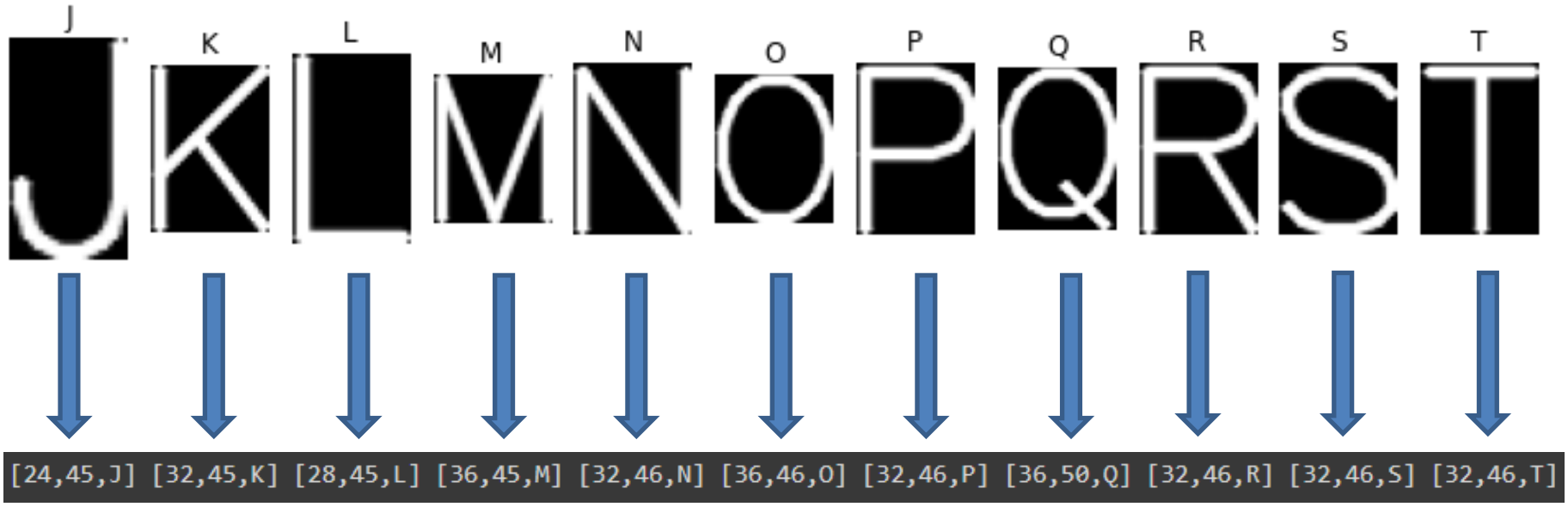






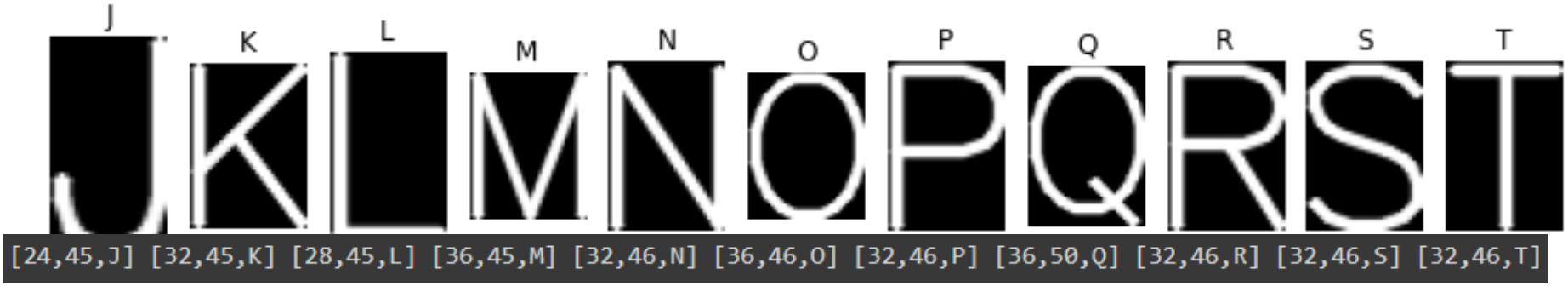
Vetor de Característica

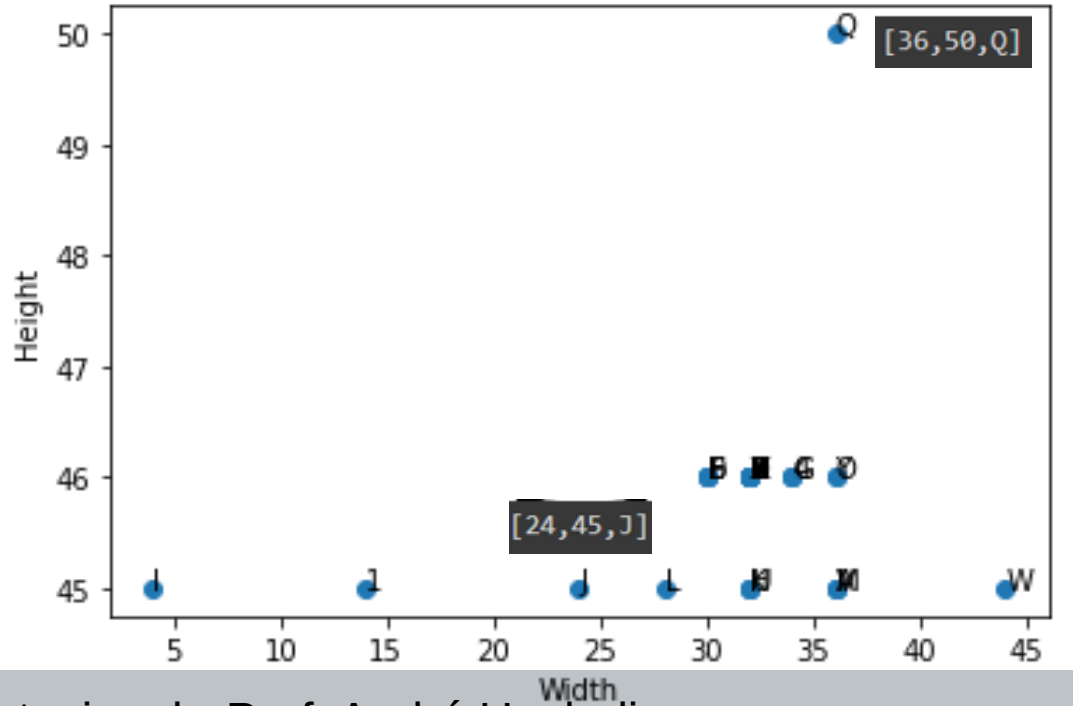
- Dada uma imagem representada pelas suas dimensões, então uma imagem que pertence a classe X é representada por:
  - f(I,X) = [I.width,I.heigth,X]



Vetor de Característica

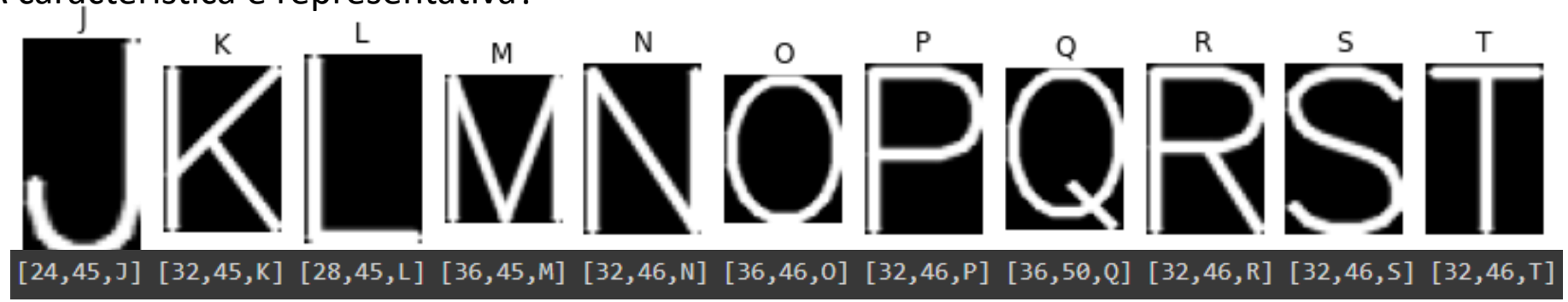
- A característica é representativa?

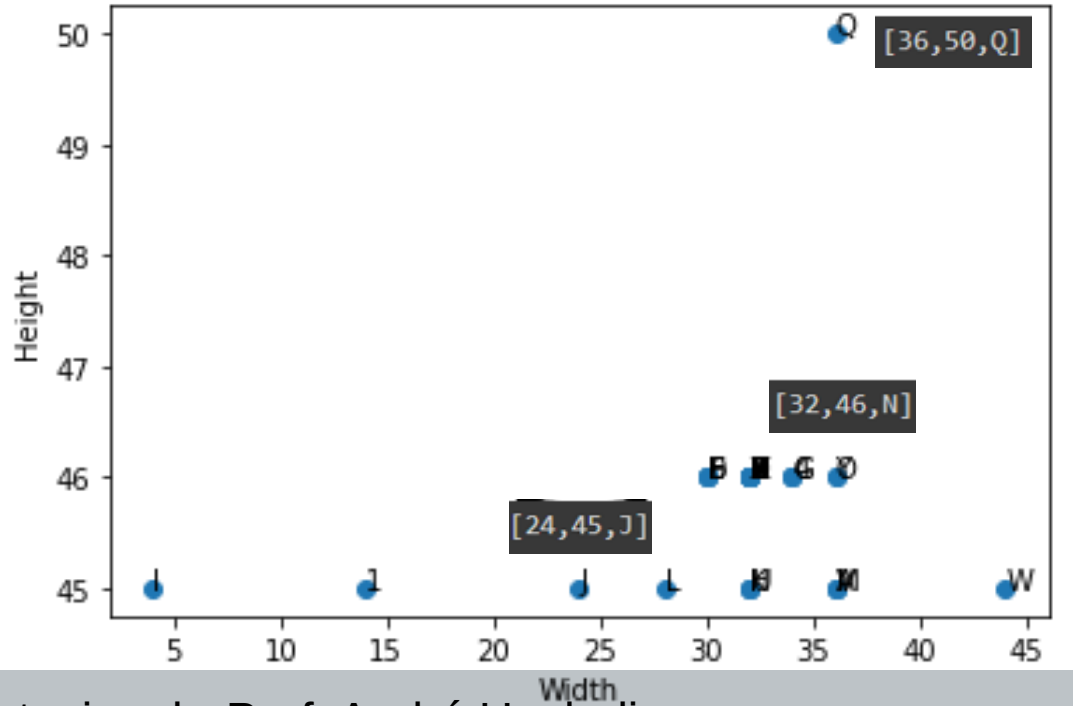




Vetor de Característica

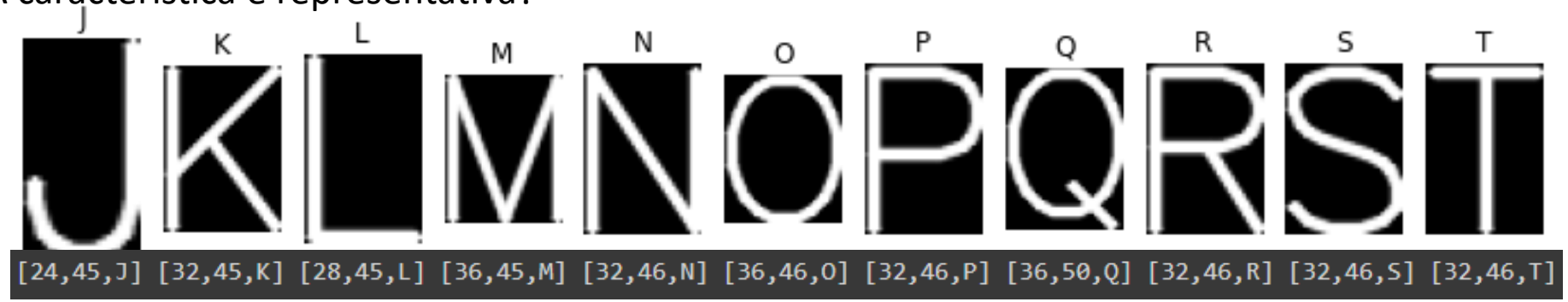
- A característica é representativa?

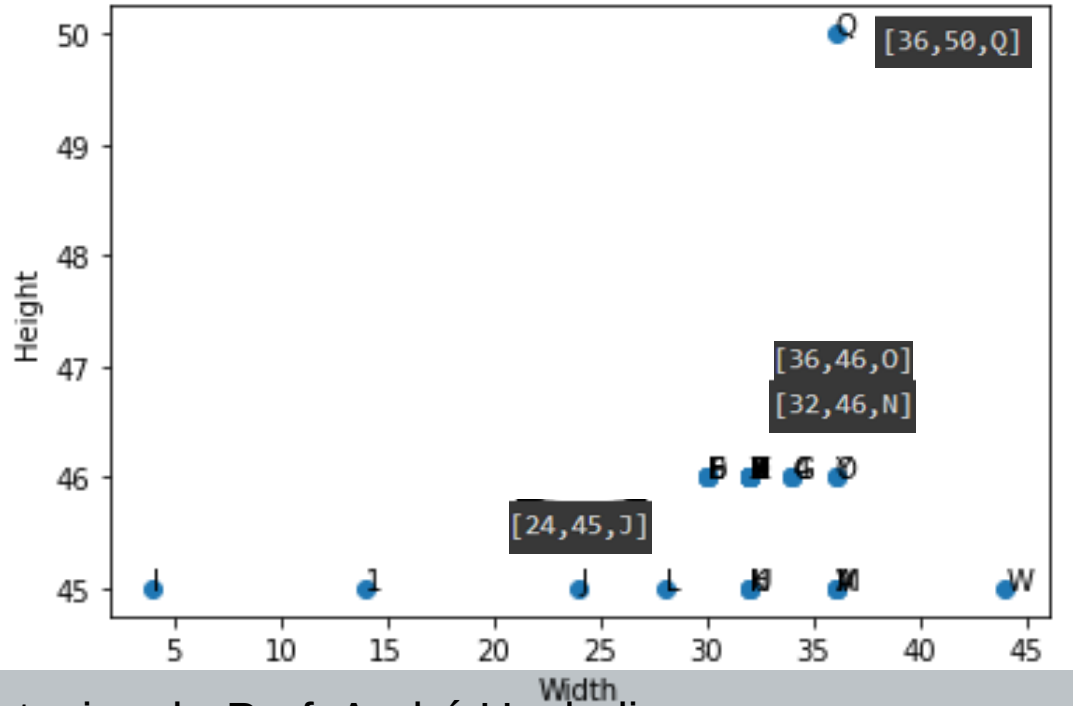




Vetor de Característica

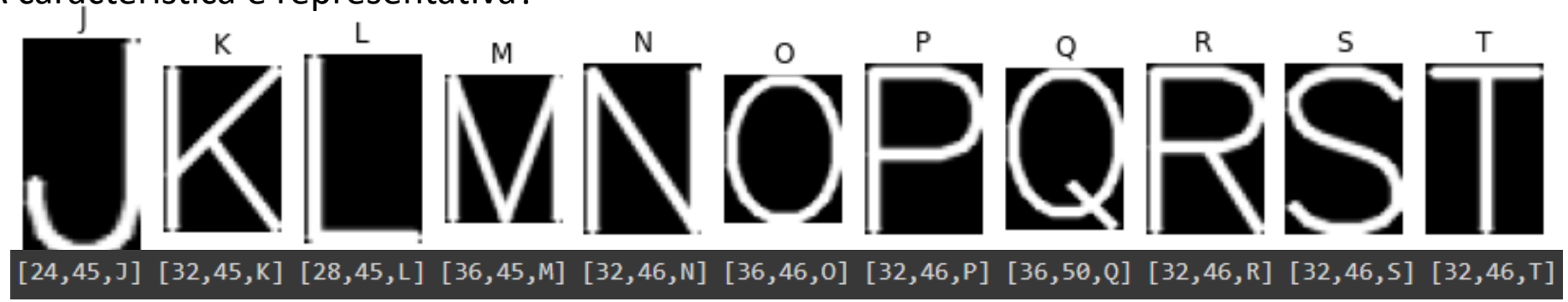
- A característica é representativa?

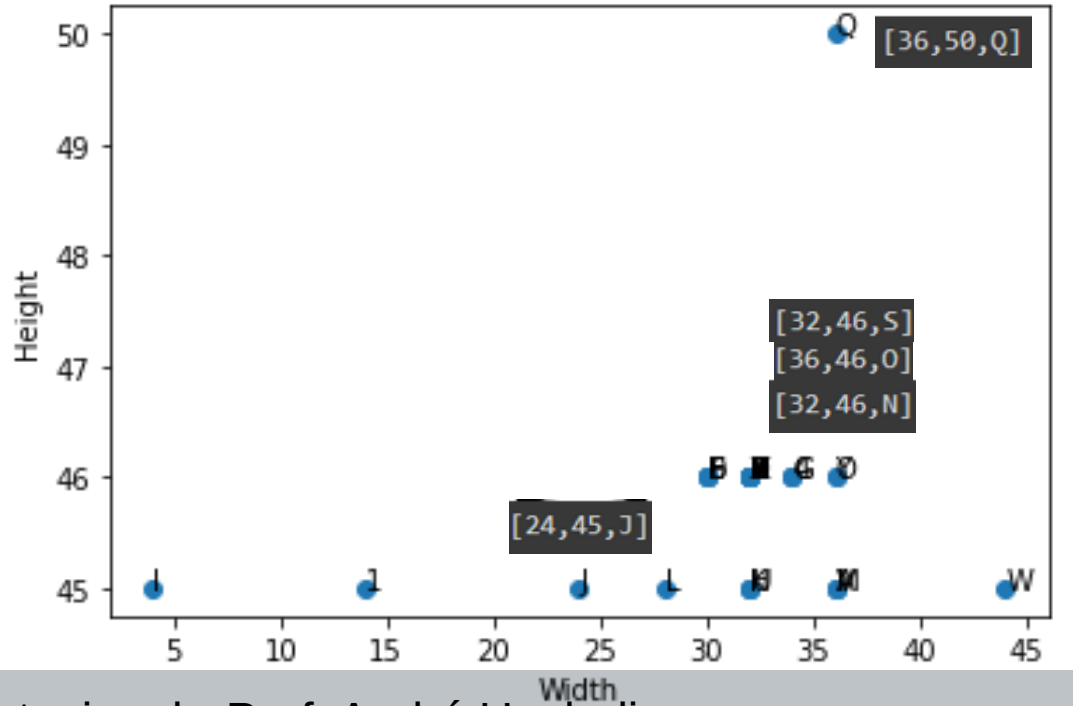




Vetor de Característica

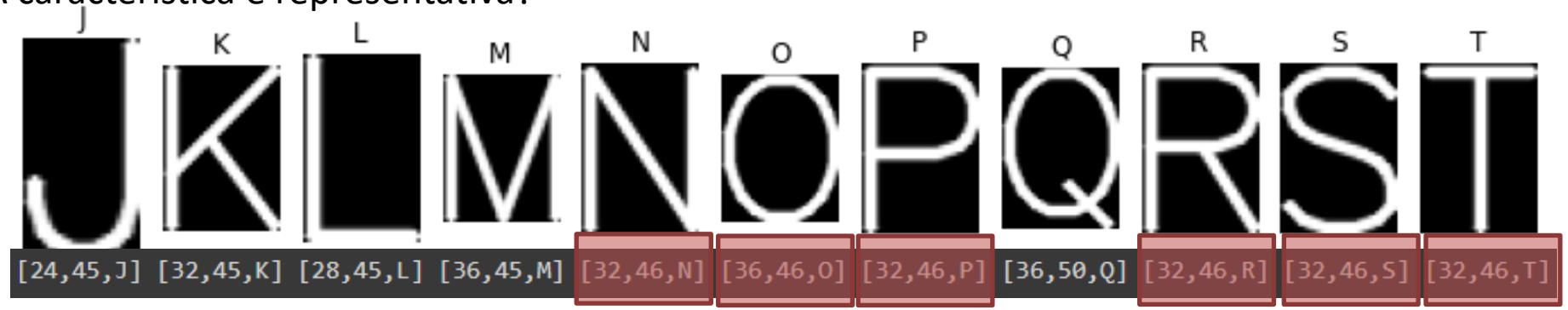
- A característica é representativa?

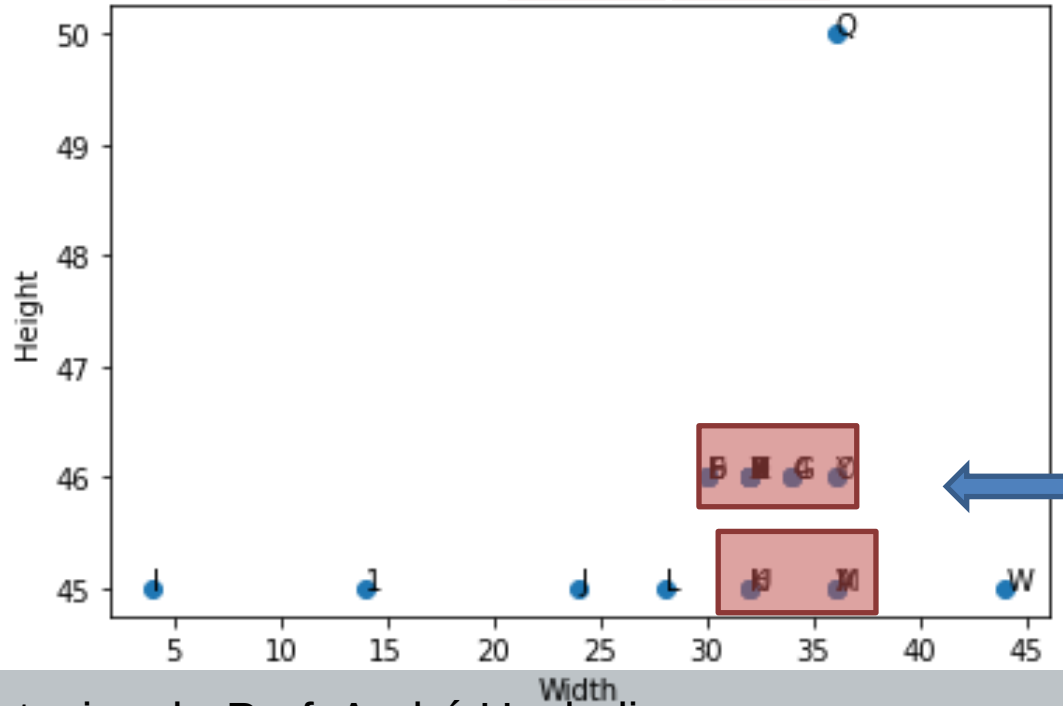




Vetor de Característica

- A característica é representativa?

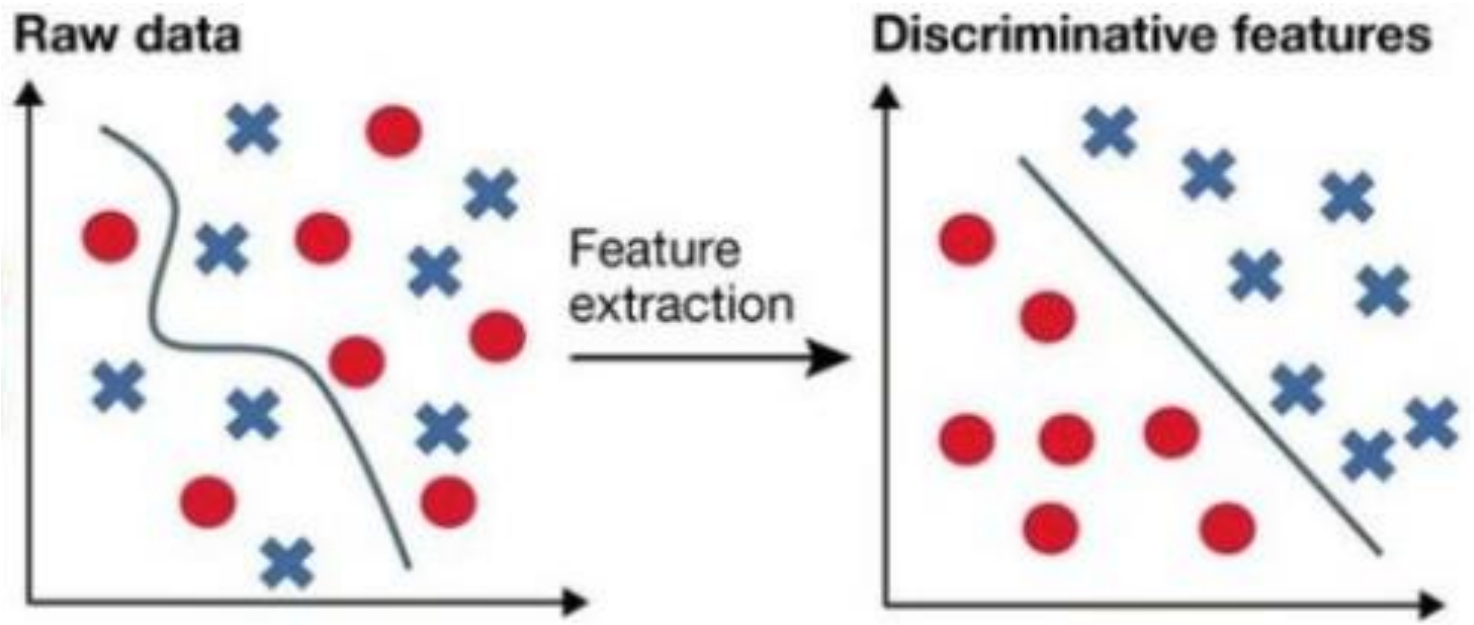




Non Discriminative

Vetor de Característica

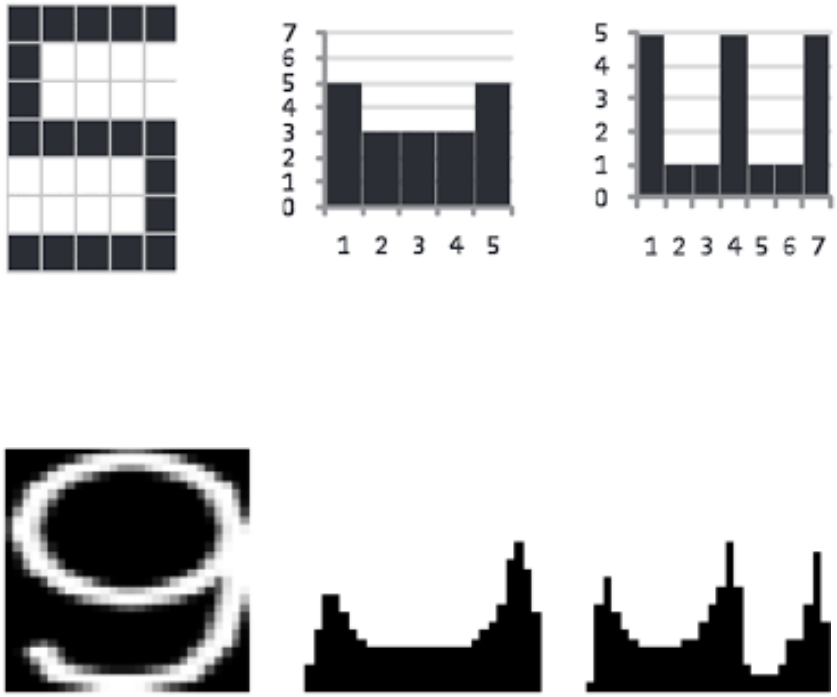
- How to produce a discriminative feature space?
- Features must describe a singular characteristic of the problem for good generalization.

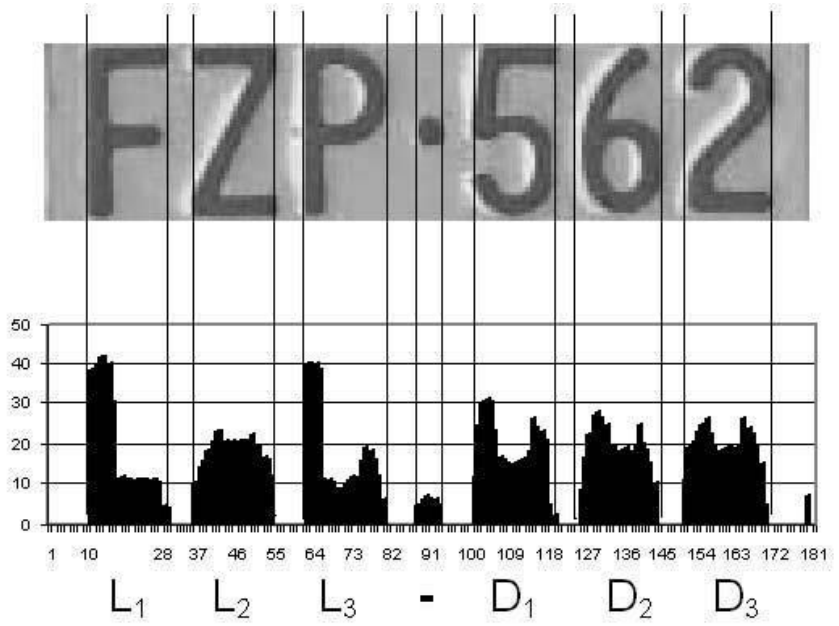


Visão Computacional - Prof. André Hochuli

Descritores

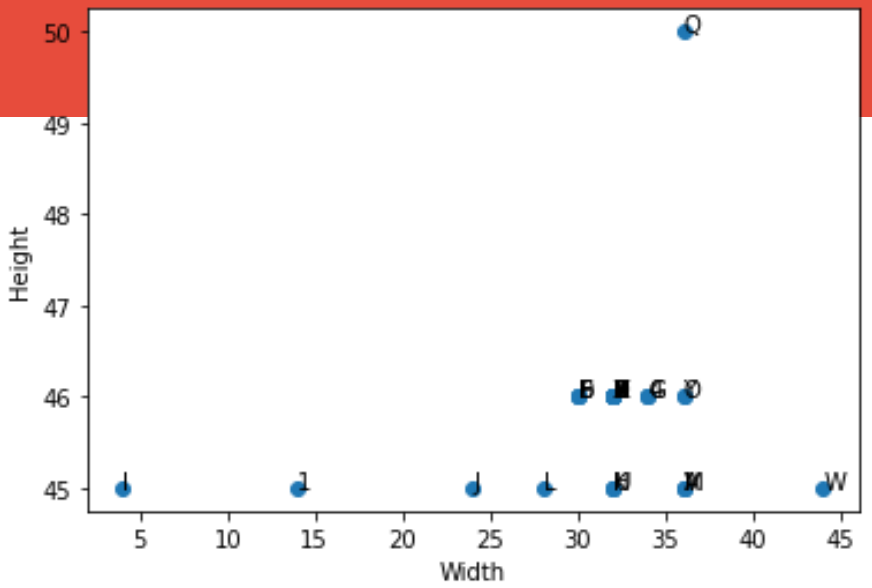
- Histogramas de Projeção Vertical e Horizontal

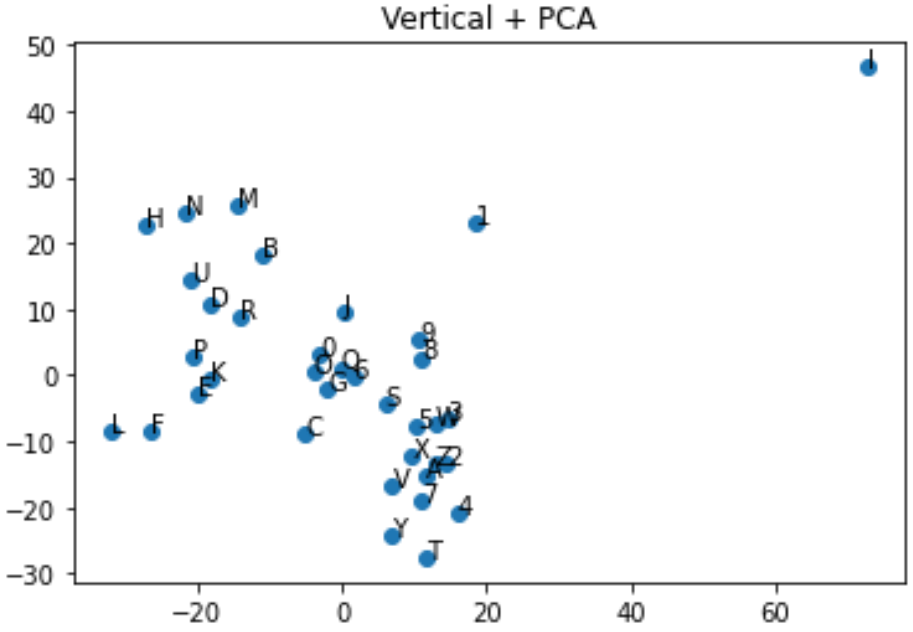


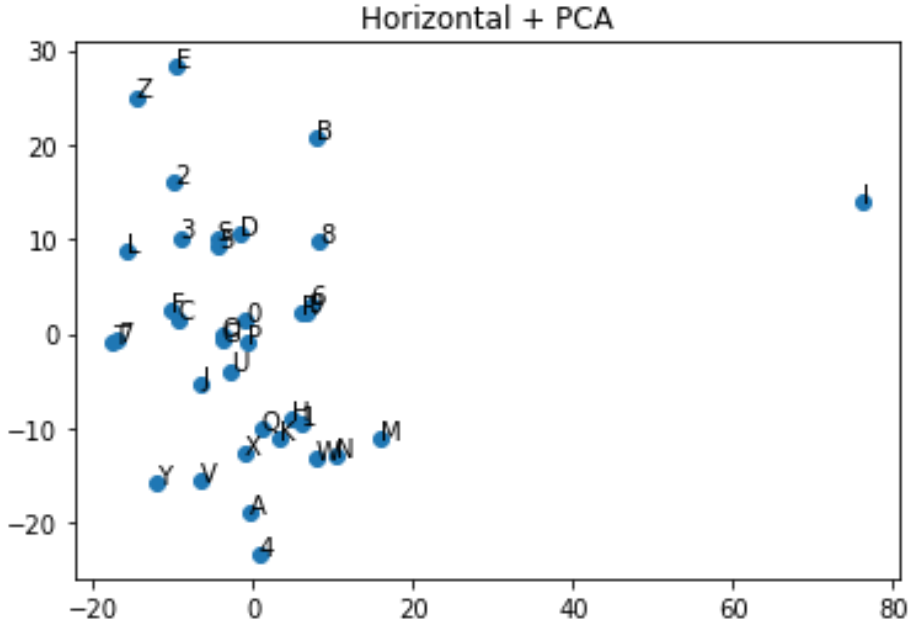


Descritores

- Histogramas de Projeção Vertical e Horizontal







Let’s Code!!

- Siga o [LINK]